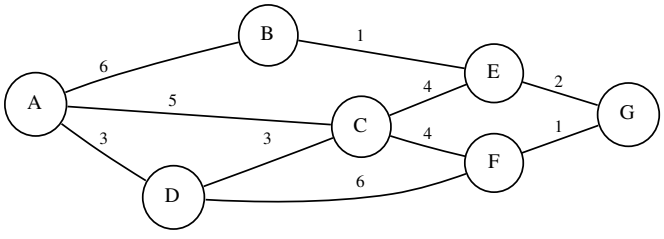
CSCI 3323
Fall 2022
Exam 3

Last Name: _____

First Name: _____

Unless instructed otherwise, write answers on the answer sheets provided. During this exam you may access a printout of the Java Command Sheet (without commentary) and a printout of an single multi-threaded program you have written.

1. (2 pts each) Name the complexity of each of these algorithms/operations.
   (a) quicksort (worst case)
   (b) heapify
   (c) radix sort
   (d) mergesort
   (e) BFS
   (f) Dijkstra's single-source shortest path algorithm
2. (6 pts) Trace by hand radix sort for this array by showing the initial order of the elements and then the order of the elements after each pass:
   $A =< 123, 323, 22, 112, 321, 133, 131, 13, 222 >$.
3. (6 pts) Trace counting sort for this array: $A =< 5, 3, 2, 5, 2, 1, 5 >$.
4. (2 pts each) Briefly define each of the following terms:
   (a) stable sort
   (b) degree of a vertex
   (c) complete graph
5. (4 pts) List two stable sorts.
6. (6 pts) Suppose a 32-bit Java integer named `set` holds a bit string representing set membership for up to 32 elements. Write a function called `getSize` that will accept the set as a parameter and will return the number of elements in the set. HINT: This requires that you count the number of 1's in the set. This can be done by using a loop to shift one at a time (either right or left) and then checking to see if the least/most-significant bit is a 1.
7. Answer the following questions based on the graph below. When tracing an algorithm, always select the node that comes first in the alphabet if the order doesn't matter to the algorithm.



(a) (4 pts) Show how the graph could be represented as a weighted adjacency matrix. Use ∞ to represent the absense of an edge.

(b) (6 pts) Trace Dijkstra's shortest path algorithm on the graph. Redraw the graph after every step and indicate the edges selected for inclusion in the shortest path by drawing them bold faced. Start with vertex `A`.

(c) (4 pts) List the order in which vertices in the graph would be visited if the graph were traversed using a DFS starting with vertex `A`.

8. There is a theorem that states:

   Theorem 1 *In a graph G with n vertices where $n \geq 3$, a Hamiltontian cycle exists if the degree of each vertex in G is at least $\frac{n}{2}$.*

   (a) (6 pts) Write a method named `hasHamiltonianCycle` that accepts (as parameters) an adjacency matrix along with the number of vertices in the graph represented by the matrix. The method should return `true` if the graph has an Hamiltonian cycle based on the rules given in the theorem above. The method should return `false` otherwise.

   (b) (2 pts) In class we claimed that there is no known simple method for determining whether or not an abitrary graph has a Hamiltonian cycle. How does that claim hold up in the presence of the above theorem?

9. Suppose Taylor county in Texas has implemented a distributed system for reporting Covid-19 cases to a central database. In this arrangement individual clinics can send their daily updates via web portal. Since there are hundreds of clinics reporting and most are reporting soon after closing hours at 5pm there have been numerous situations where concurrent updates have led to incorrect results. You have been tasked with updating the backend code so that concurrent transactions will be properly handled. Study the existing backend code below and then answer these questions. NOTE: The VirusStats object is designed to be shared among multiple threads.

   (a) (6 pts) Write a section of code that would go in a driver that will create and run 500 StatRecorder threads and then display the summary statistics after the threads have been completed.

   (b) (6 pts) Use one or more semaphores to ensure that all provided data will be properly recorded even if multiple entries are reported at exactly the same time. Your use of semaphores should maximize concurrency while guaranteeing correct results. Write your modifications to the code on the exam sheet.

```
class StatRecorder implements Runnable {
   private VirusStats stats;

   public StatRecorder(VirusStats vs) {
      stats= vs;
   }

   public void run() {
      Scanner kb= new Scanner(System.in);
      int [] localResults= new int[3];

      System.out.println("How many tests did your clinic administer: ");
      localResults[0]= kb.nextInt();
      System.out.println("How many tests were positive: ");
      localResults[1]= kb.nextInt();
      System.out.println("How many recoveries: ");
      localResults[2]= kb.nextInt();

      stats.reportResults(localResults);
   }
}

class VirusStats {
   private int [] stats;

   public VirusStats() {
      stats= new int[3];
   }

   public void display() {
      System.out.println("Tests Administered: " + stats[0]);
      System.out.println("Positive Results  : " + stats[1]);
      System.out.println("Negative Results  : " + stats[0] - stats[1]);
      System.out.println("Recoveries        : " + stats[2]);
      System.out.println("Net Change        : " + stats[1] - stats[2]);
   }

   public void reportResults(int [] results) {
      for (int i=0; i<3; i++) {
         stats[i]+= results[i];
      }
   }
}
```